Probability Distribution Modifications by the “luck” Role Buff in Skarmbot

+52 Feb 05

ArgoTheNaut

Table of Contents

Problem Definition

Context

The discord bot contains a leveling system that rewards members for active participation. By default, experience is rewarded to members at most once per minute for sending a message in the server. The amount of experience gained for a single message can range from 15 to 25, inclusive. This can be described as the equivalent statement 15 + $\lfloor X \rfloor$ where X is a random variable distributed uniformly. That is,

$$X \sim U(0,11)$$

To create a more dynamic (read: pay to win) reward system, four key areas have been identified where experience rewards can be modified. They are as follows:

- Base experience gain (15) [b]
- Random bonus maximum (10) [r]
- Cooldown between messages that qualify for experience rewards (60 seconds)
- Random bonus probability distribution, "luck" (g(X)) [U]

The objective for all four parameters is to have a buff modifier to the parameter that is initially zero, ideally provides roughly linear improvements per point of the buff, and can be set to any non-negative value less than 1 million (which would achieve the top level in a mere six minutes) without causing problems, such as a divide by zero error would.

The first two features can easily be given linear modifiers (15 + n, 10 + m), and the cooldown between messages can be given a linear buff per cooldown point using the formula below, explained further at the source (LoL: Ability Haste):

$$\text{Reduced cooldown} = \text{Base Cooldown} \times \frac{100}{100 + \text{Haste}}$$

Given conditions

Following a similar formula for the "luck" modifier for the otherwise uniform distribution of the random variable. Using the following transformation:

$$U = g(X) = X^a \qquad a = \frac{100}{100 + luck} \qquad 0 \le luck \le 1{,}000{,}000 \qquad X \sim U(0,1)$$

X is now distributed uniformly from 0 to 1. Multiplication by the random bonus maximum is evaluated after the calculation.

Reasoning behind use of the transformation U = X^p

The family of functions $f(x) = x^p$ has a useful property in a probabilistic application wherein the input values of x 0 and 1 remain as constant outputs for all values of p greater than zero. That is to say,

$$\forall p \in R^+ : (0^p = 0 \cap 1^p = 1)$$

Due to the domain of the starting uniform distribution being from 0 to 1, the continuous codomain being constrained to between these two points proves useful for knowing the endpoints of the CDF will always be 0 and 1 at the input extrema of 0 and 1. As demonstrated in the following graphs, a value of $p > 1$ produces a decrease in the output for any input x between 0 and 1 while a value of $p < 1$ produces an increase in the output for any input x between 0 and 1.

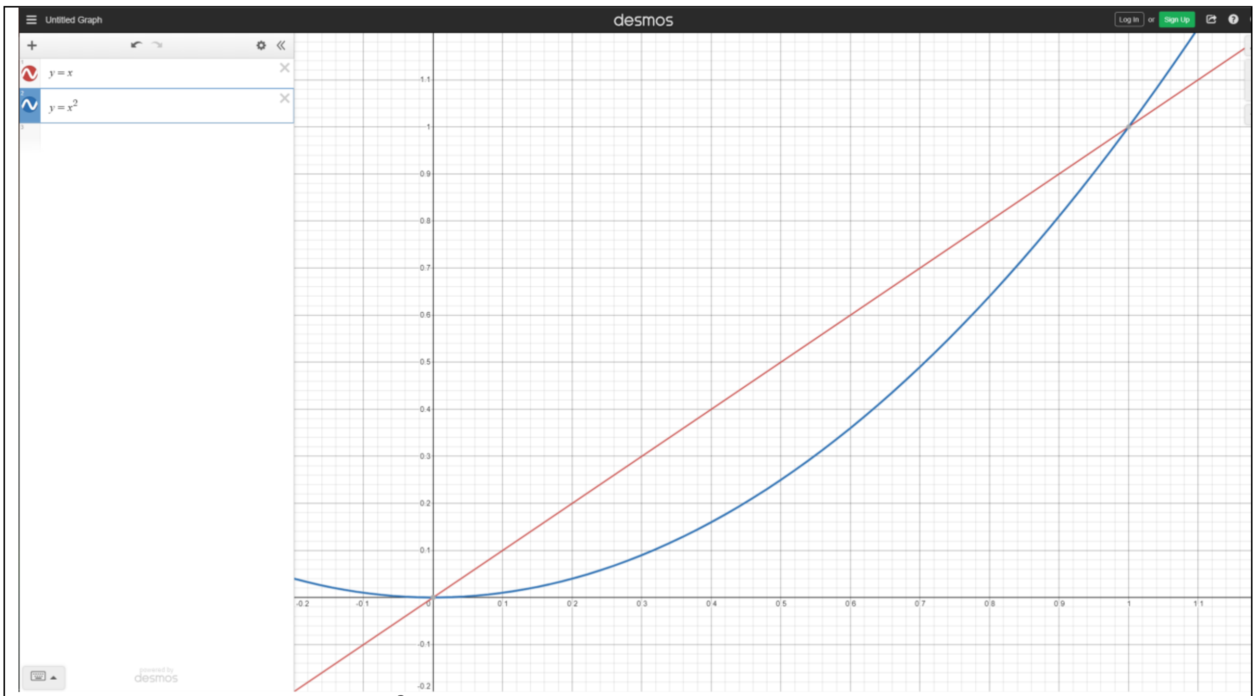

Fig. 1: The transformation $y = x^2$ produces lower output values for $0 < x < 1$ compared to $y = x$.
https://www.desmos.com/calculator/ayzecx5vxk

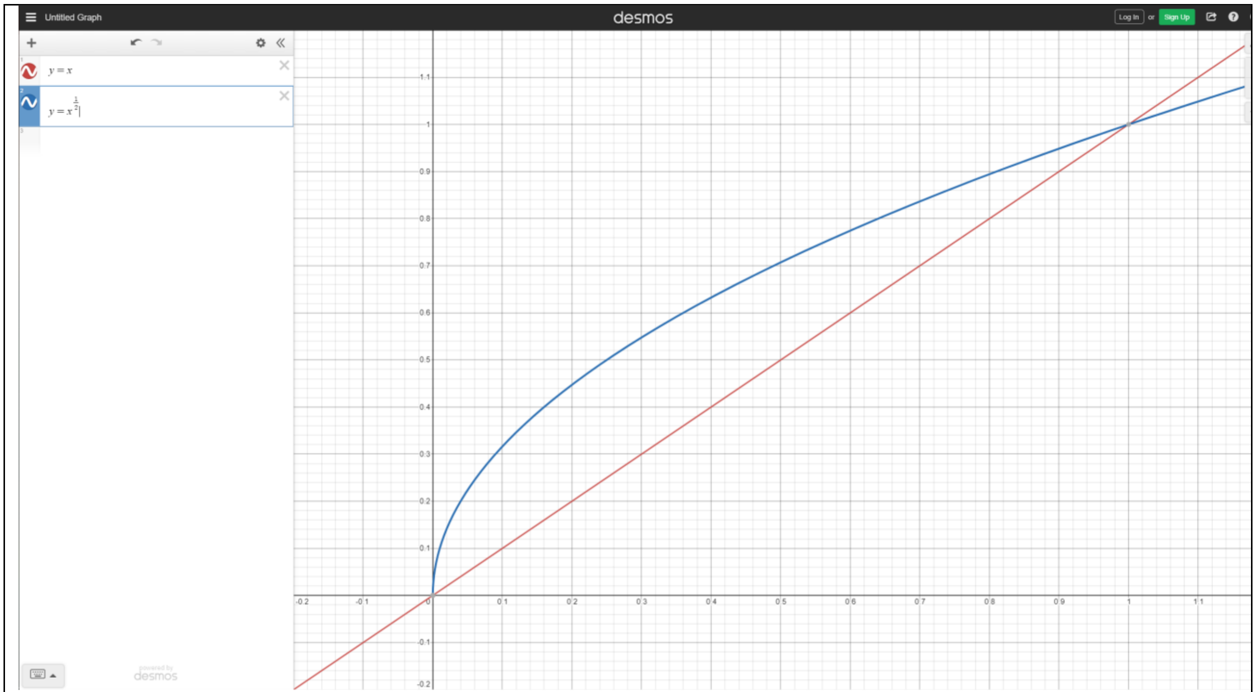

Fig. 2: The transformation $y=x^{0.5}$ produces greater output values for $0 < x < 1$ compared to $y = x$.
https://www.desmos.com/calculator/7if3aim2tm

Because of these properties, $U = X^p$ becomes a good candidate function for the purposes of modifying probability. However, due to the default value of the luck buff for any given role being 0 and expecting a linear increase, another function must be applied to *p* in order to sanitize it as a safe function to use. For this purpose, the function p = 100 / (100 + luck) is selected.

For a default luck value of 0, p evaluates to 1. This value of p offers no transformation to the experience gain function and can be considered a good non-harmful base. For any value of luck greater than 0, p will asymptotically approach 0, and all values of x^p will approach 1. This is the desired asymptotic relationship for the luck stat.

The use of 100 in the numerator and denominator sets this as the magnitude at which luck buffs should be set in order to see significant change. This also leaves 99 free intermediate integers that can be used as micro-buffs without needing to resort to fractional values.

While this is not the only way by which luck can be effectively established, it is the method used in this analysis.

Analysis

Using the CDF technique knowing that $f_x(x) = \begin{cases} 1, & 0 < x < 1 \\ 0, & otherwise \end{cases}$

$$F_U(u) = P(U \le u) = P(X^a \le u) = P\left(X \le u^{\frac{1}{a}}\right) = F_X\left(u^{\frac{1}{a}}\right)$$

$$f_U(u) = \frac{d}{du}\big(F_U(u)\big) = f_X\left(u^{\frac{1}{a}}\right) * \frac{d}{du}\left(u^{\frac{1}{a}}\right) = 1 * \frac{1}{a} * u^{(\frac{1}{a}-1)}$$

For this particular case, the useful information to have is the expected value of the random variable as a function of the user's luck.

$$E[U] = \int_{u=0}^{u=1} u * f_U(u)du = \int_{u=0}^{u=1} \frac{u^{(1/a)}}{a} du = \left. \frac{u^{(1+\frac{1}{a})}}{(1+\frac{1}{a})a} \right|_{u=0}^{u=1} = \frac{1}{a+1}$$

Substituting in the value of a with the function of luck defined above,

$$E[U] = \frac{1}{a+1} = \frac{1}{\frac{100}{100+luck}+1} = \frac{1}{\frac{200+luck}{100+luck}} = \frac{100+luck}{200+luck} = 1 - \frac{100}{200+luck}$$

With these calculations, the expected value of the random variable over the range of luck from 0 (no modifiers) to 1000 (maximum bonus from a single role) is as follows:

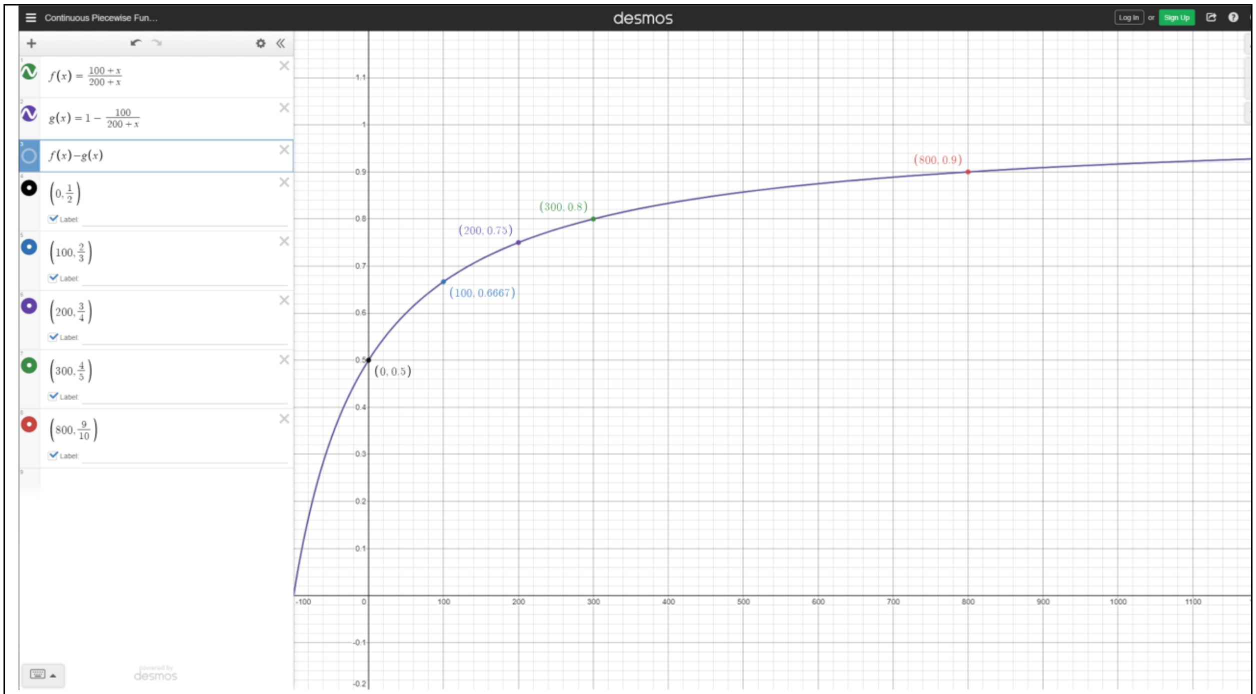

Fig. 3: the expected value of the random variable modified by the luck of a role.
https://www.desmos.com/calculator/jyeaiydkhg

The expected value acts as a long term projection of the effectiveness of the random bonus maximum and its modifier.

With the new distribution defined by U, the experience gained for any qualifying message, after modifications as applicable, becomes

$$EXP = b + r * U$$

Appendix A: Individual distribution graphs

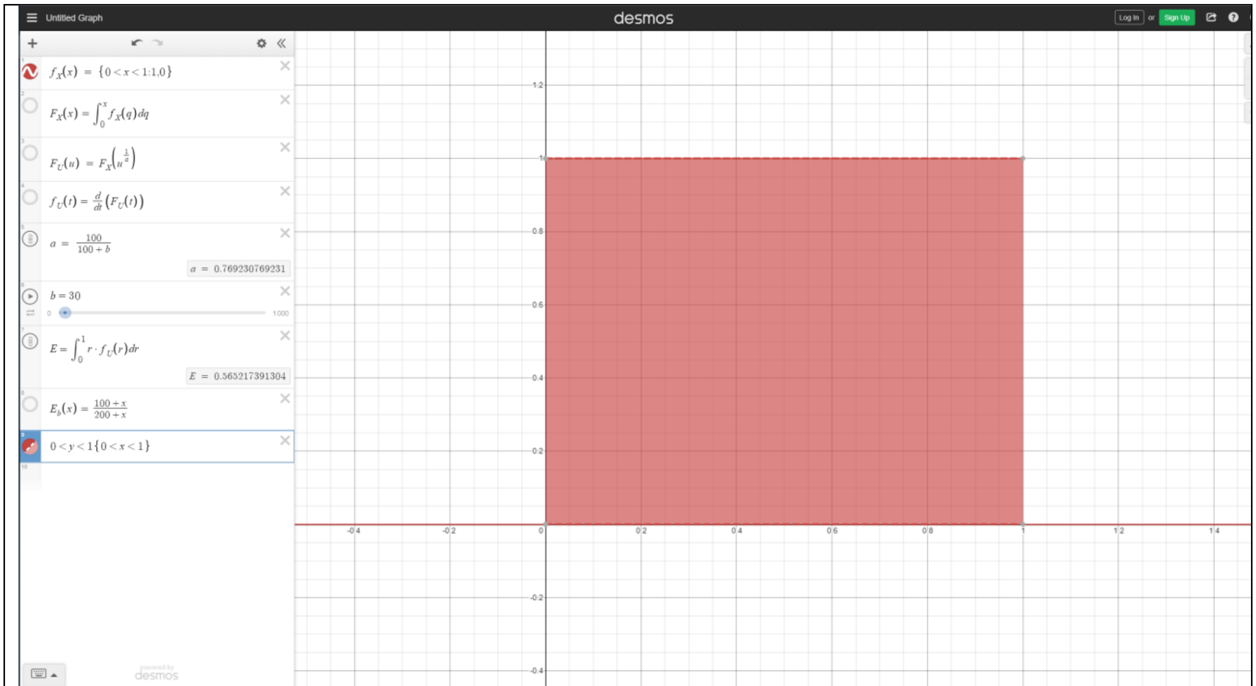

Fig. 4: The probability density function of X~U(0,1)
https://www.desmos.com/calculator/ke18tlgjin

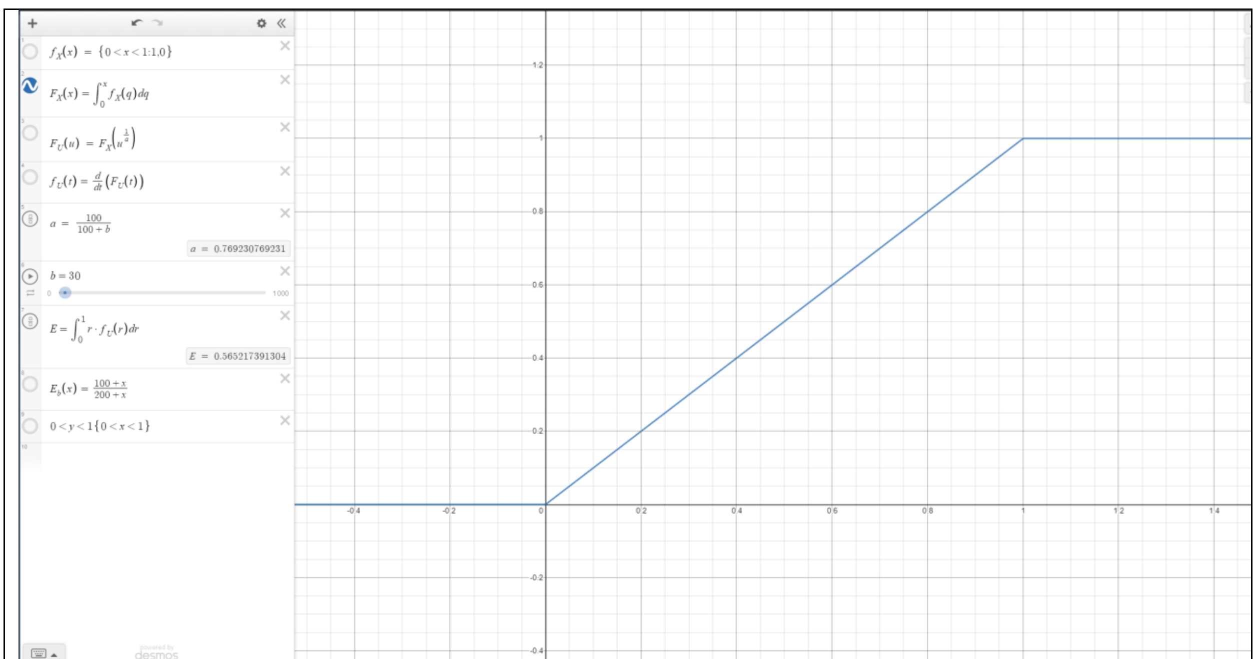

Fig. 5: The Cumulative Distribution Function of X~U(0,1)
https://www.desmos.com/calculator/vtjb5rpoai

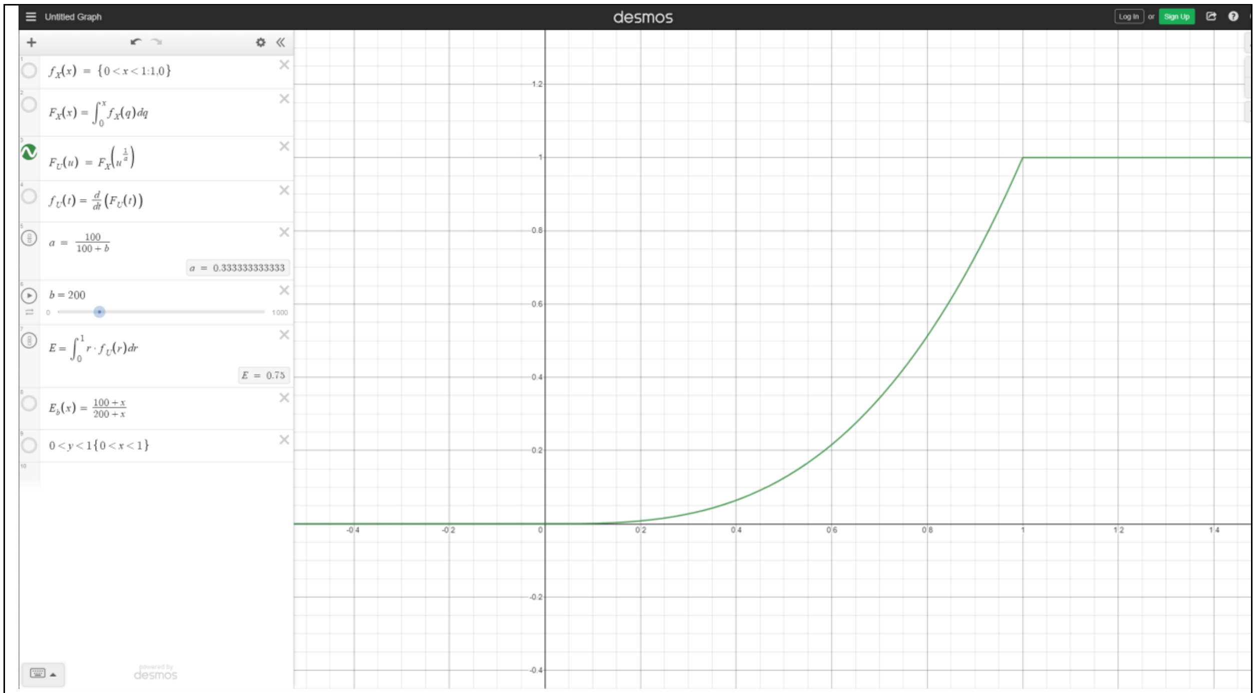

Fig. 6: The Cumulative Distribution Function of U = X^a for a luck value of 200 → a=1/3
https://www.desmos.com/calculator/ux3tjsajpx

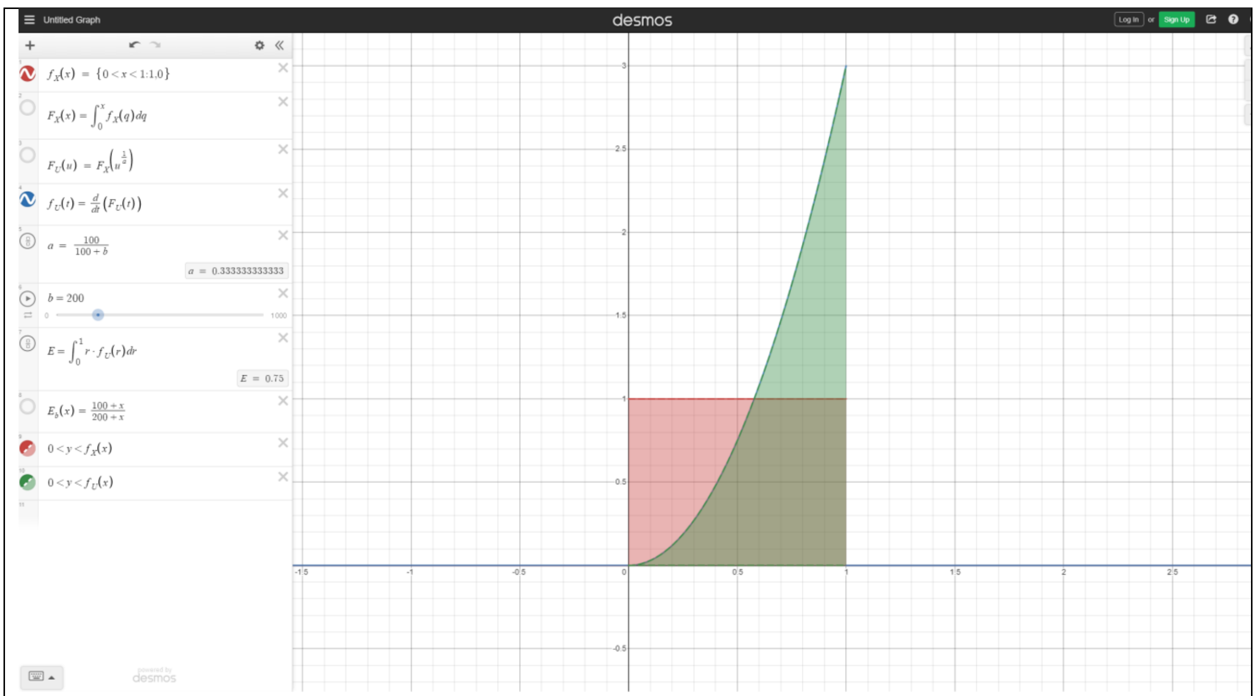

Fig. 7: The probability density functions for X (red, distributed uniformly) and U (green, the transformation of X). Notice that a significantly greater portion of the area under the curve appears to the left of 0.5 in the distribution of U.
https://www.desmos.com/calculator/3qaegg5nld

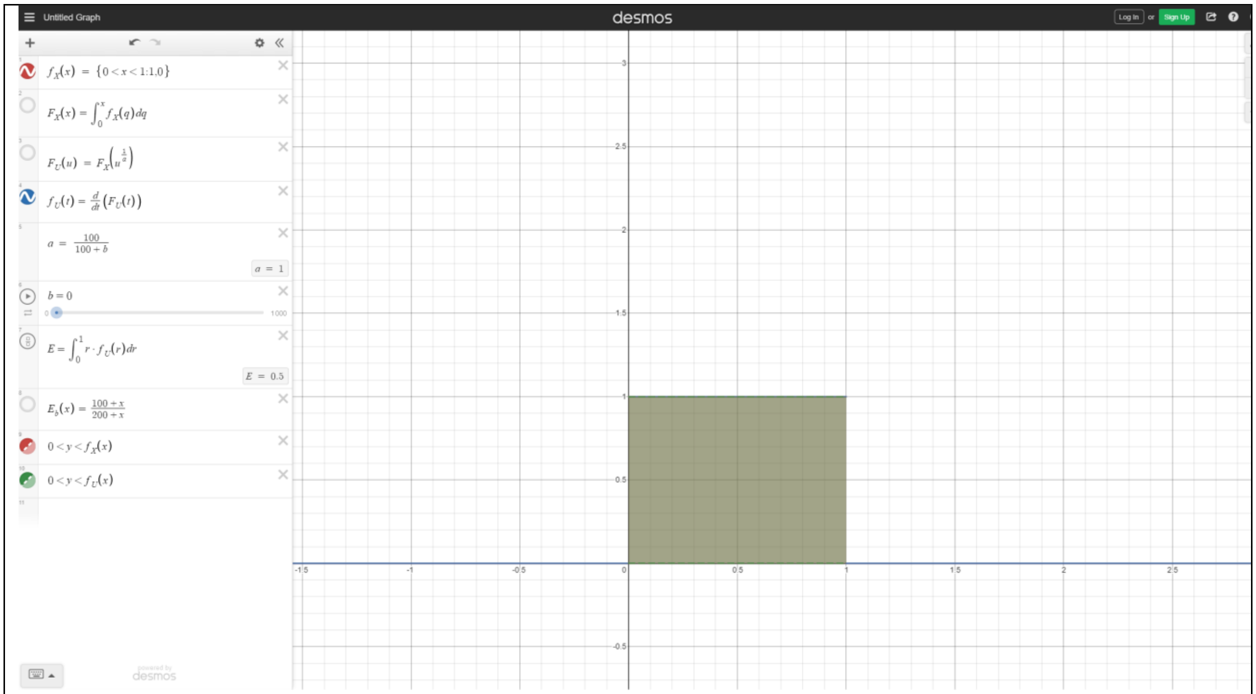

Fig. 8: When the luck bonus is set to 0, the transformation makes no change to the uniform distribution, preserving the same experience for guilds that don't chose to use this feature. https://www.desmos.com/calculator/zhsrmjdnff

Appendix B: Empirical programming verification (node.js)

Program

```
// Beginning of program
"use strict";
const TRIAL_COUNT  = 10000;
function getExpectedValue(LUCK){
	let EV = 0;
	//returns a random value between 0 and 1
	function randResult(){
		return Math.pow(Math.random(), (100)/(100 + LUCK));
	}
	for(let i = 0; i < TRIAL_COUNT; i++){
		EV += randResult();
	}
	return EV / TRIAL_COUNT;
}
for(let luck = 0; luck < 1001; luck += 100){
	let EV = getExpectedValue(luck);
	console.log(`Expected value given luck =\t${luck}:\t${EV}\tIdeal:
${(100+luck)/(200+luck)}`);
}
// End of program
```

Execution output

```
Console output:
Expected value given luck =	0:	0.4966735510681596	Ideal: 0.5
Expected value given luck =	100:	0.6709645102270652	Ideal: 0.6666666666666666
Expected value given luck =	200:	0.7479804802867689	Ideal: 0.75
Expected value given luck =	300:	0.8014302848633157	Ideal: 0.8
Expected value given luck =	400:	0.8332899158503493	Ideal: 0.8333333333333334
Expected value given luck =	500:	0.8586513842725044	Ideal: 0.8571428571428571
Expected value given luck =	600:	0.8762311459099273	Ideal: 0.875
Expected value given luck =	700:	0.888499559802101	Ideal: 0.8888888888888888
Expected value given luck =	800:	0.9004658949275961	Ideal: 0.9
Expected value given luck =	900:	0.9091978481747085	Ideal: 0.9090909090909091
Expected value given luck =	1000:	0.9166335114831361	Ideal: 0.9166666666666666
```